IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	Neil A. COOPER			
Title:	SYSTEM FOR LOADING DEVICE-SPECIFIC CODE AND METHOD THEREOF			
App. No.:	09/904,989	Filed:		July 13, 2001
Examiner:	Diem K. CAO	Group Art Unit:	2194	
Customer No.:	34456	Confirmation No.:	3444	
Atty. Dkt. No.:	ATI.0100820 (1376-0100820)			

Mail Stop: Appeal Brief- Patents
The Board of Patent Appeal and Interferences
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

Adam D. Sheehan, Reg. No. 42,146
Attorney for Appellant
LARSON NEWMAN ABEL POLANSKY & WHITE, L.L.P.
(512) 439-7100 (phone)
(512) 439-7199 (fax)

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. § 41.37(c)(1)):

TABLE OF CONTENTS

The final page of this brief before the beginning of the Appendix of Claims bears the attorney's signature.

I.   REAL PARTY IN INTEREST (37 C.F.R. § 41.37(c)(1)(i))

The real party in interest in this appeal is ATI Technologies, Inc.

II.   RELATED APPEALS AND INTERFERENCES (37 C.F.R. § 41.37(c)(1)(ii))

There are no interferences or other appeals that will directly affect, or be directly affected by, or have a bearing on the Board's decision in this appeal.

III.   STATUS OF CLAIMS (37 C.F.R. § 41.37(c)(1)(iii))

A.   TOTAL NUMBER OF CLAIMS IN APPLICATION

There are thirty-five (35) claims pending in the application (claims 1 and 3-36).

B.   STATUS OF ALL THE CLAIMS

1.   Claims pending:

Claims 1 and 3-36.

2.   Claims withdrawn from consideration but not canceled:

NONE.

3.   Claims allowed:

23-30.

4.   Claims objected to:

8, 9, 15, 17, and 32-34.

5.   Claims rejected:

Claims 1, 3-7, 10-14, 16, 18-22, 31, and 35-36 are rejected under 35 U.S.C. § 103.

6. Claims canceled:

Claim 2.

C. CLAIMS ON APPEAL

There are three (4) claims on appeal, claims 1, 13, 16, and 31.

IV. STATUS OF AMENDMENTS (37 C.F.R. § 41.37(c)(1)(iv))

There were no amendments filed subsequent to final rejection of the claims.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER (37 C.F.R. § 41.37(c)(1)(v))

The following summary is provided to give the Board the ability to quickly determine where the claimed subject matter appealed herein is described in the present application and is not to limit the scope of the claimed invention.

The subject matter of independent claim 1 is directed to a method. Claim 1 recites the features of loading device-independent driver code into kernel mode memory (e.g., element 122, FIG.1), wherein the device-independent driver code forms a first portion (e.g. element 132, FIG.1) of a display driver (e.g. element 130, FIG.1, block 230, FIG. 2, and p. 8, lines 19-22). Claim 1 also recites the features of requesting a device identifier (e.g. element 167, FIG.1) after loading the device-independent driver code into kernel mode memory, wherein the requested device identifier is to identify a particular device (e.g. block 240, FIG. 2, and p. 8, lines 23-25). Additionally, the method recites the features of receiving the requested device identifier associated with a particular device (e.g. p. 8, lines 25-26). Claim 1 further recites the features of identifying a particular device-specific driver portion (e.g. element 144, FIG. 1) from a plurality of driver portions associated with the device identifier based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an

- 2 -

application program interface (e.g. block 250, FIG. 2, and p. 8, line 26 – p. 9, line 1). In addition, claim 1 recites the features of loading the particular device-specific driver portion into kernel mode memory, wherein the device-specific driver portion forms a second portion of the display driver (e.g. element 134, FIG. 1, block 280, FIG. 2, and p. 9, lines 9-11).

The subject matter of independent claim 13 is directed to a method. Claim 13 recites the features of providing a set of device-independent functions (e.g. element 132, FIG.1), wherein the device-independent functions are capable of manipulating a processor (e.g. element 110, FIG.1) to support a plurality of different display devices (e.g. block 420, FIG. 4, p. 11, lines 9-12). Claim 13 also recites the features of providing a plurality of device-specific driver portions (e.g., elements 144 and 146, FIG. 1), wherein each device-specific driver portion of the plurality of device-specific driver portions include functions capable of manipulating a processor to support only a portion of the plurality of different display devices (e.g. block 440, FIG. 4, p. 11, lines 17-19). Claim 13 additionally recites the features of providing a first function to manipulate a processor to load one or more device-independent functions (e.g. element 134, FIG. 1) of the set of device-independent functions into kernel mode memory (e.g. block 470, FIG. 4, p. 11, line 26 – p. 12, line 6). Claim 13 further recites the features of providing a second function to manipulate a processor to request for a device identifier (e.g. element 167, FIG. 1) after the one or more device-independent functions are loaded into kernel mode memory, wherein the device identifier is capable of identifying a particular display device of the plurality of different display devices (e.g. block 470, FIG. 4, p. 11, line 26 – p. 12, line 6). In addition, claim 13 recites the features of providing a table linking device identifiers to individual device-specific driver portions of the plurality of device-specific driver portions and providing a third function to manipulate a processor to load a particular device-specific driver portion into kernel

mode memory, wherein the particular device-specific driver portion is associated with the particular display device of the plurality of different display devices. (e.g. block 470, FIG.4, p. 11, line 26 – p. 12, line 6).

Independent claim 31 is a device claim. Claim 31 recites the features of computer readable medium tangibly embodying a plurality of programs of instructions, including a set of device-independent functions to manipulate a processor (e.g. element 110, FIG. 1) to support a plurality of different display devices (e.g. element 160, FIG. 1, block 420, FIG. 4, p. 11, lines 9-12). Claim 31 further recites the features of a plurality of device-specific driver portions, wherein each device-specific driver portion of the plurality of device-specific driver portions includes functions to manipulate a processor to support only a portion of the plurality of different display devices (e.g. block 440, FIG. 4, p. 11, lines 17-19). Claim 31 also recites the features of a first function to manipulate a processor to load one or more device-independent functions (e.g. element 132, FIG. 1) of the set of device-independent functions into kernel mode memory (e.g. block 440, FIG. 4, p. 11, lines 18-21). Claim 31 additionally recites the features of a second function to manipulate a processor to request a device identifier (e.g. element 167, FIG. 1) after the one or more device-independent functions of the set of device-independent functions into kernel mode memory, wherein the device identifier is capable of identifying a particular display device of the plurality of different display devices (e.g. block 460, FIG. 4, p. 11, lines 25-27). Claim 31 further recites the features of a third function to manipulate a processor to identify a particular device-specific driver by locating a name associated with the particular device-specific driver portion in a table using the device identifier and a fourth function to manipulate a processor to load a particular device-specific driver portion (e.g., element 134, FIG. 1) into kernel mode memory, wherein the particular device-specific driver portion is associated with the

particular display device of the plurality of different display devices (e.g. block 470, p. 11, line 26 – p. 12, line 6).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL (37 C.F.R. § 41.37(c)(1)(vi))

A. Claims 1, 13, and 31 are rejected under 35 U.S.C. § 103(a) as being unpatentable over *Bondy* et al. (U.S. Patent No. 5,491,813) (hereinafter "the *Bondy* reference") in view of *Keller* et al. (U.S. Patent No. 5,752,032) (hereinafter "the *Keller* reference") and further in view of *Schoening* et al. (U.S. Patent No. 6,226,788) (hereinafter "the *Schoening* reference") as set forth in the Final Office Action dated June 5, 2006 (hereinafter, "the Final Action") and the subsequent Advisory Action dated August 8, 2006 (hereinafter, "the Advisory Action").

B. Claim 16 is rejected under 35 U.S.C. § 103(a) as being unpatentable over the *Bondy* reference in view of the *Keller* reference. further in view of *Shirakabe* et al. (U.S. Patent No. 5,136,709 (hereinafter "the *Shirakabe* reference") as set forth in the Final Action and the subsequent Advisory Action.

VII. ARGUMENTS (37 C.F.R. § 41.37(c)(1)(vii))

Based on the arguments and issues below, none of the claims stand or fall together, because in addition to having different scopes, each of the independent claims has a unique set of issues relating to its rejection and appeal as indicated in the arguments below:

    A.   Rejection of Claims 1, 13, and 31 under 35 U.S.C. § 103 (37 C.F.R. §1.192(c)(8)(iv)):

On page 2 of the Final Action, claims 1, 13, and 31 were rejected under 35 U.S.C. § 103(a) as unpatentable over the *Bondy* reference in view of the *Keller* reference and further in view of the *Schoening* reference. Claims 1, 13 and 31 are reproduced below for ease of reference.

1. (Previously Presented) A method comprising:
   loading device-independent driver code into kernel mode memory, wherein the device-independent driver code forms a first portion of a display driver;
   requesting a device identifier after loading the device-independent driver code into kernel mode memory, wherein the requested device identifier is to identify a particular device;
   receiving the requested device identifier associated with a particular device;
   identifying a particular device-specific driver portion from a plurality of driver portions associated with the device identifier based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface; and
   loading the particular device-specific driver portion into kernel mode memory, wherein the device-specific driver portion forms a second portion of the display driver.

13. (Previously Presented) A method comprising:

provides a set of device-independent functions, wherein the device-independent functions are capable of manipulating a processor to support a plurality of different display devices;

providing a plurality of device-specific driver portions, wherein each device-specific driver portion of the plurality of device-specific driver portions include functions capable of manipulating a processor to support only a portion of the plurality of different display devices;

providing a first function to manipulate a processor to load one or more device-independent functions of the set of device-independent functions into kernel mode memory;

providing a second function to manipulate a processor to request for a device identifier after the one or more device-independent functions are loaded into kernel mode memory, wherein the device identifier is capable of identifying a particular display device of the plurality of different display devices; and

providing a table linking device identifiers to individual device-specific driver portions of the plurality of device-specific driver portions;

providing a third function to manipulate a processor to load a particular device-specific driver portion into kernel mode memory based on the table and the device identifier, wherein the particular device-specific driver portion is associated with the particular display device of the plurality of different display devices.

31. (Previously Presented) A computer readable medium tangibly embodying a plurality of programs of instructions, the plurality of programs including:

a set of device-independent functions to manipulate a processor to support a plurality of different display devices;

a plurality of device-specific driver portions, wherein each device-specific driver portion of the plurality of device-specific driver portions includes functions to manipulate a processor to support only a portion of the plurality of different display devices;

a first function to manipulate a processor to load one or more device-independent functions of the set of device-independent functions into kernel mode memory;

a second function to manipulate a processor to request a device identifier after the one or more device-independent functions of the set of device-independent functions into kernel mode memory;

a third function to manipulate a processor to identify a particular device-specific driver by locating a name associated with the particular device-specific driver portion in a table using the device identifier; and

a fourth function to manipulate a processor to load the particular device-specific driver portion into kernel mode memory.

According to 35 U.S.C. § 103(a), "[a] patent may not be obtained though the invention is

not identically disclosed or described as set forth in section 102 of this title, if the differences

between the subject matter sought to be patented and the prior art of such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains."

In *ex parte* examination of patent applications, the Patent Office bears the burden of establishing *a prima facie* case of obviousness. *In re Fritch*, 972 F.2d 1260,1262,23 U.S.P.Q. 2d 1780, 1783 (Fed. Cir. 1992). The initial burden of establishing a *prima facie* basis to deny patentability to a claimed invention is always upon the Patent Office. *In re Oetiker,* 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Piasecki,* 745 F.2d 1468, 1472, 223 U.S.P.Q. 785, 788 (Fed. Cir. 1984). Only when a *prima facie* case of obviousness is established does the burden shift to the applicant to produce evidence of nonobviousness. *In re Oetiker,* 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Rijckaert, 9* F.3d 1531, 1532, 28 U.S.P.Q.2d 1955, 1956 (Fed. Cir. 1993). If the Patent Office does not produce *a prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker,* 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak,* 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985).

A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell,* 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make

the claimed invention and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure.

The Final Action asserts that the proposed combination of the *Bondy* reference, the *Keller* reference, and the *Schoening* reference disclose the claimed invention as recited by claims 1, 13, and 31. The Appellant disagrees and submits that the proposed combination of the *Bondy* reference, the *Keller* reference, and the *Schoening* reference fails to disclose or suggest each and every limitation recited by claims 1, 13, and 31. The Final Action admits at page 4 and 5 that the *Bondy* reference and the *Keller* reference fail to disclose or suggest "device driver management, including locating a name associated with the device-specific driver portion in a table using the device identifier (device type value)" and relies on the *Schoening* reference to disclose these features. However, as explained below, the *Schoening* reference in fact fails to disclose or suggest those elements of claims 1, 13, and 31 lacking in the *Bondy* reference and the *Keller* reference.

1) The *Schoening* Reference Fails To Disclose Identifying A Particular Device-Specific Driver Portion

Claim 1 recites, in part, the limitations of identifying a particular device-specific driver portion from a plurality of driver portions associated with the device identifier based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface. The Appellant submits that the *Schoening* reference fails to disclose or suggest identifying a device specific driver portion based on a comparison of versions associated with functions of the device specific driver portion to versions expected through an application program interface as recited in claim 1.

- 9 -

With regard to these limitations, the Final Action asserts that the device mapping table (e.g. the Device Mapper 1214) of the *Schoening* reference identifies device-specific driver based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface. *Final Action*, p. 4. However, the Appellant submits that the *Schoening* reference fails to disclose or suggest identifying device specific driver portions, much less identifying device-specific driver based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface as alleged by the Final Action. Instead, the *Schoening* reference discloses identifying functions *of an application program interface* to be overridden.

In particular, the *Schoening* reference discloses a "Device Mapper",

> As shown in FIG. 2C, a Device Mapper 1214 is associated with each device 102 or device type. Each Device Mapper 1214 is associated with a list 216 of overridden functions. Each entry in the list 216 identifies a Service Module Function 76a-76n that is overridden by the Device Mapper 1214 for its associated device or device type. Each Device Mapper 1214 is also associated with a list 1218 of identifiers of devices that are managed by the Device Mapper.

The *Schoening* reference, col. 13, lines 59-66. Accordingly, the Device Mapper is a table associated with a device or device type in a network that identifies Service Module Functions that are overridden for the associated device or device type. *Id.* The Service Module Functions identified by the Device Mapper are "overridden functions of a service module." *Id.* at col. 6, lines 64-65. Further, a service module is "a set of classes derived from the FrameWork and FrontEnd packages *that define the API*, data model, database, and abstract functions that implement network device services." *Id.* at col. 6, lines 60-63 (emphasis added). The FrameWork disclosed in the *Schoening* reference "means the set of classes, in an object-oriented computer programming language, and services from which the organization and structure of a

Service module is derived. In particular, a FrameWork d*efines the structure of an API* and internal dispatch mechanisms." *Id.*, col. 6, lines 42-46 (emphasis added).

In addition, the *Schoening* reference indicates

The Handle Versions method 358 provides a mechanism to locate a Device Mapper based not only on OID but also on a version identifier. In standard MIB variables, software version information is not included. Therefore, a separate mechanism is required to enable the ANI 50 to match a particular version of hardware, software, or firmware of a device 102 to a Device Mapper. Preferably, entries in a Device Mapper may include information that specifies a range of versions over to which the Device Mapper applies. For example, the entries may provide version brackets indicating that a particular Device Mapper is chosen only within a particular range of versions, or from a particular version and all subsequent versions, etc.

Id. at col. 17, lines 16-28. Thus, the system disclosed in the *Schoening* reference can determine which Device Mapper applies for a particular device version.

Accordingly, the Device Mapper of the *Schoening* reference identifies, for a particular device version a set of overridden functions of a set of classes that define an application program interface (API). The set of classes is derived from a framework that defines the structure of the API. Thus, the *Schoening* reference discloses disabling certain functions of a device by overriding functions of a set of classes associated with the device from an *application program interface*. There is no disclosure or suggestion in the *Schoening* reference that the set of classes, or their functions, are device-specific driver portions for at least the reason that the *Schoening* reference fails to teach that they are used to control the operation of a specific peripheral device in any manner. Instead, the *Schoening* reference states that the set of classes "*define the API, data model, database, and abstract functions* that implement network device services." *Id.* at col. 6, lines 60-63 (emphasis added). Accordingly, the Device Mapper identifies portions of an API and associated *abstract functions* of network device services, rather than *device specific driver*

portions. One of skill in the art will appreciate that an API is an interface between application programs and an operating system, while a driver portion control functions of a device. Accordingly, the Device Mapper does not identify device-specific driver portions, but instead portions of an interface between application programs and an operating system. Therefore, the *Schoening* reference fails to disclose or suggest identifying a particular *device-specific driver portion* from a plurality of driver portions associated with the device identifier *based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface* as recited in claim 1.

<div align="center">

2) The Proposed Combination of the *Bondy, Keller* and *Schoening*
    References Does Not Disclose or Suggest Each and Every Limitation of
    the Claims Under Appeal

</div>

As noted above, the *Schoening* reference fails to disclose or suggest identifying a particular device-specific driver portion from a plurality of driver portions associated with the device identifier based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface, as recited in claim 1.

With respect to claim 13, the claim recites providing a third function to manipulate a processor to load a particular device-specific driver portion into kernel mode memory, wherein the particular device-specific driver portion is associated with the particular display device of the plurality of different display devices. As explained above, the *Schoening* reference fails to disclose or suggest identifying or loading device-specific driver portions, as recited in claim 13.

With respect to claim 31, the claim a third function to manipulate a processor to identify a particular device-specific driver by locating a name associated with the particular device-specific driver portion in a table using the device identifier. As explained above the *Schoening*

reference fails to disclose or suggest identifying device-specific driver portions as recited in claim 31.

Further, as admitted in the Final Office Action at page 4, the *Keller* reference and the *Bondy* reference fail to disclose "device driver management" including the limitations of claims 1, 13, and 31 that are lacking in the Schoening reference. As such, the proposed combination of the cited references fails to disclose or suggest each and every limitation of claims 1, 13, and 31 and the Final Action therefore has failed to establish that the proposed combination of the cite references discloses or suggests each and every claim depending from claims 1, 13, or 31 at least by virtue of this dependency.

B.   Rejection of Claim 16 under 35 U.S.C. § 103 (37 C.F.R. §1.192(c)(8)(iv)):

On page 6 of the Final Action, claim 16 was rejected under 35 U.S.C. § 103(a) as unpatentable over the *Bondy* reference in view of the *Keller* reference and further in view of the *Shirakabe* reference. Claim 16 depends from claim 13, and thus includes all the limitations of claim 13. However, in order to reject claim 13, the Final Action relied upon the *Schoening* reference, which is not relied upon to reject claim 16. Accordingly, by its own terms, the Final Action should rely upon the *Schoening* reference to support a prima facie case of obviousness as to claim 16. However, as pointed out above the Final Action does not in fact rely upon the *Schoening* reference to reject claim 16. Accordingly, the Final Action by its own terms fails to show that each and every element of claim 16 is disclosed or suggested by the *Bondy*, *Keller*, or *Shirakabe* references. The Final Action therefore fails to establish a *prima facie* case of obviousness in support of its rejection of claim 16.

VIII. CONCLUSION

For the reasons given above, the Appellant respectfully requests reconsideration and allowance of all claims and that this patent application be passed to issue.

Respectfully submitted,

<u>March 12, 2007</u>
Date

/Adam D. Sheehan/
Adam D. Sheehan, Reg. No. 42,146
Attorney for Appellant
LARSON NEWMAN ABEL POLANSKY & WHITE, L.L.P.
(512) 439-7100 (phone)
(512) 439-7199 (fax)

IX. **APPENDIX OF CLAIMS INVOLVED IN THE APPEAL (37 C.F.R. § 41.37(c)(1)(viii))**

The text of each claim involved in the appeal is as follows:

1. (Previously Presented) A method comprising:
loading device-independent driver code into kernel mode memory, wherein the device-independent driver code forms a first portion of a display driver;
requesting a device identifier after loading the device-independent driver code into kernel mode memory, wherein the requested device identifier is to identify a particular device;
receiving the requested device identifier associated with a particular device;
identifying a particular device-specific driver portion from a plurality of driver portions associated with the device identifier based on a comparison of versions associated with functions of the device-specific driver portion to versions expected through an application program interface; and
loading the particular device-specific driver portion into kernel mode memory, wherein the device-specific driver portion forms a second portion of the display driver.

13. (Previously Presented) A method comprising:
providing a set of device-independent functions, wherein the device-independent functions are capable of manipulating a processor to support a plurality of different display devices;
providing a plurality of device-specific driver portions, wherein each device-specific driver portion of the plurality of device-specific driver portions include functions capable of manipulating a processor to support only a portion of the plurality of different display devices;
providing a first function to manipulate a processor to load one or more device-independent functions of the set of device-independent functions into kernel mode memory;
providing a second function to manipulate a processor to request for a device identifier after the one or more device-independent functions are loaded into kernel mode memory, wherein the device identifier is capable of identifying a particular display device of the plurality of different display devices; and
providing a table linking device identifiers to individual device-specific driver portions of the plurality of device-specific driver portions;
providing a third function to manipulate a processor to load a particular device-specific driver portion into kernel mode memory based on the table and the device identifier, wherein the particular device-specific driver portion is associated with the particular display device of the plurality of different display devices.

16. (Previously Presented) The method as in Claim 13, further including providing a fourth function to determine addresses associated with functions of the particular device-specific driver portion, after providing the third function.

31. (Previously Presented) A computer readable medium tangibly embodying a plurality of programs of instructions, the plurality of programs including:

    a set of device-independent functions to manipulate a processor to support a plurality of different display devices;

    a plurality of device-specific driver portions, wherein each device-specific driver portion of the plurality of device-specific driver portions includes functions to manipulate a processor to support only a portion of the plurality of different display devices;

    a first function to manipulate a processor to load one or more device-independent functions of the set of device-independent functions into kernel mode memory;

    a second function to manipulate a processor to request a device identifier after the one or more device-independent functions of the set of device-independent functions into kernel mode memory;

    a third function to manipulate a processor to identify a particular device-specific driver by locating a name associated with the particular device-specific driver portion in a table using the device identifier; and

    a fourth function to manipulate a processor to load the particular device-specific driver portion into kernel mode memory.

32. (Original) The computer readable medium as in Claim 31, wherein the second function includes a call to an EngLoadImage function.

33. (Original) The computer readable medium as in Claim 32, further including a third function to determine addresses associated with functions of the particular device-specific driver portion.

34. (Original) The computer readable medium as in Claim 33, wherein the third function includes a function call to an EngFindImageProcAddress function.

35. (Original) The computer readable medium as in Claim 31, wherein the device identifier includes an application specific integrated circuit identifier.

36. (Original) The computer readable medium as in Claim 31, further including a table linking device identifiers to individual device-specific driver portions of the plurality of device-specific driver portions.

X.   EVIDENCE APPENDIX (37 C.F.R. § 41.37(c)(1)(ix))

None.

XI. **RELATED PROCEEDINGS APPENDIX (37 C.F.R. § 41.37(c)(1)(x))**

None.